1.

Job Sequencing Problem

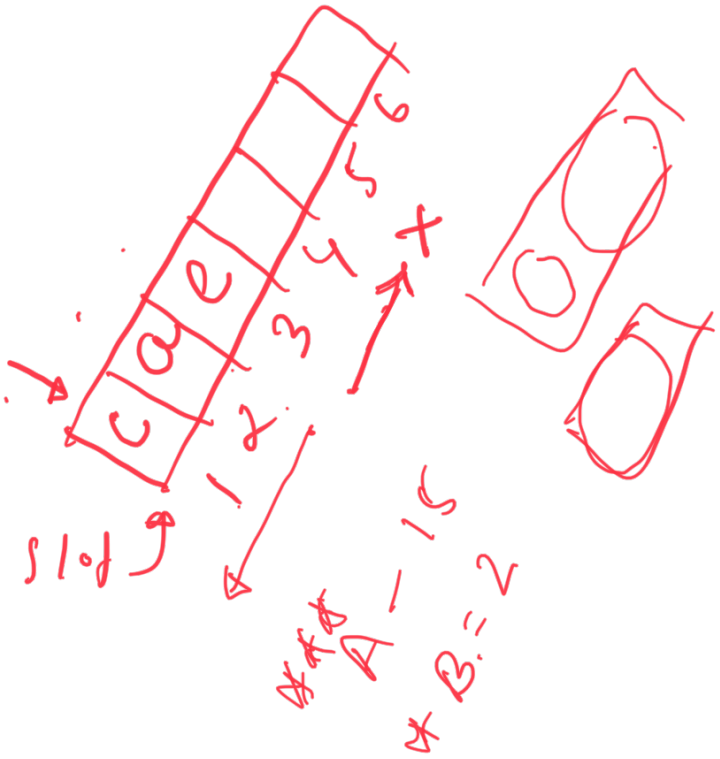
Given an array of jobs where every job has a deadline and associated profit if the job is finished before the deadline. It is also given that every job takes the single unit of time, so the minimum possible deadline for any job is 1. How to maximize total profit if only one job can be scheduled at a time.

Input: Four Jobs with following deadlines and profits

JobID	Deadline	Profit
a	4	20
b	1	10
c	1	40
d	1	30

Output: Following is maximum profit sequence of jobs

c, a



Input: Five Jobs with following deadlines and profits

JobID	Deadline	Profit
a	2	100
b	1	19
c	2	27
d	1	25
e	3	15

Output: Following is maximum profit sequence of jobs

c, a, e

A Simple Solution is to generate all subsets of given set of jobs and check individual subset for feasibility of jobs in that subset. Keep track of maximum profit among all feasible subsets. ***The time complexity of this solution is exponential***. This is a standard Greedy Algorithm problem.

Following is the algorithm.

1) Sort all jobs in decreasing order of profit.
2) Iterate on jobs in decreasing order of profit. For each job , do the following :
   a) Find a time slot i, such that slot is empty and i < deadline and i is greatest. Put the job in This slot and mark this slot filled.
   b) If no such i exists, then ignore the job.

2.

Job Sequencing Problem – Loss Minimization

We are given N jobs numbered 1 to N. For each activity, let Ti denotes the number of days required to complete the job. For each day of delay before starting to work for job i, a loss of Li is incurred.
We are required to find a sequence to complete the jobs so that overall loss is minimized. We can only work on one job at a time.

Input : L = {3, 1, 2, 4} and
T = {4, 1000, 2, 5}
Output : 3, 4, 1, 2
Explanation: We should first complete
Job 3, then jobs 4, 1, 2 respectively.

Input : L = {1, 2, 3, 5, 6}
T = {2, 4, 1, 3, 2}
Output : 3, 5, 4, 1, 2
Explanation: We should complete jobs
3, 5, 4, 1 and then 2 in this order.

Let us consider two extreme cases and we shall deduce the general case solution from them.

1. All jobs take same time to finish, i.e. Ti = k for all i. Since all jobs take same time to finish we should first select jobs which have large Loss (Li). We should select jobs which have the highest losses and finish them as early as possible. Thus this is a greedy algorithm. Sort the jobs in descending order based on Li only.
2. All jobs have the same penalty. Since all jobs have the same penalty we will do those jobs first which will take less amount of time to finish. This will minimize the total delay, and hence also the total loss incurred. This is also a greedy algorithm. Sort the jobs in ascending order based on Ti. Or we can also sort in descending order of 1/Ti.

From the above cases, we can easily see that we should sort the jobs not on the basis of Li or Ti alone. Instead, we should sort the jobs according to the ratio Li/Ti, in descending order.

3.

Minimum Number of Platforms Required for a Railway

Given arrival and departure times of all trains that reach a railway station, the task is to find the minimum number of platforms required for the railway station so that no train waits. We are given two arrays which represent arrival and departure times of trains that stop.

Input: arr[] = {9:00, 9:40, 9:50, 11:00, 15:00, 18:00}
dep[] = {9:10, 12:00, 11:20, 11:30, 19:00, 20:00}
Output: 3
Explanation: There are at-most three trains at a time (time between 11:00 to 11:20)

	Input: arr[] = {9:00, 9:40} dep[] = {9:10, 12:00} Output: 1 Explanation: Only one platform is needed.

```
struct job
{
    int jobno;
    int profit, deadline;
} joblist[100];

int slot[100];

Sort (joblist. profit)

for (i=0; i<n; i++) {
    for (j= joblist[i]. deadline ; j>=1 ;j--)
        if (slot[j] == -1) {
            slot[j] = i;
            break;
        }
}
```